DISC OBJECT MODULE

LIBRARY MAINTENANCE

ROUTINE FOR TDOS

DOMLMR

TABLE OF CONTENTS

Title: Disc Object Module Library Maintenance Routine (DOMLMR)

Author: William R. Jones - RCA
Cleveland District
600 Independence Towers
5755 Granger Road
Cleveland, Ohio 44130
216/398-5100

Date: September 21, 1970

Description: DOMLMR maintains an Object Module Library (OML), on a 70/564 or 70/590 disc unit, in a format acceptable to the RCA Linkage Editor (LNKEDT) utility. DOMLMR will add, delete, and replace individual or groups of modules without the need to replace the entire disc resident library. Because of a difference in format between a disc OML created by the Call Library Transcriber Routine (CLTR) and DOMLMR, this utility will not maintain the master OML. Any attempt to update the master OML with DOMLMR will result in an error message and program termination.

The disc area used by DOMLMR must begin at cylinder 1, track zero, and may extend thru cylinder 200 on a 70/564 or 70/590. The minimum allocation is two cylinders. After allocation of the disc area with the RCA utility Random Access Storage Allocator (RAALLR), the area is preformatted by the PREOML program provided with this package. The allocated disc area may be increased at any time by running RAALLR, followed by PREOML.

The input to DOMLMR is an OML - the output from the Object Module Library Update routine (OMLU). Three standard OMLU control cards may be used to convert an Object Module File (OMF) from a language translator (SYSUT1) to an OML. These three control cards will convert a single or multiple module SYSUT1.

Deletion of modules from the disc OML is accomplished via a console message. The program will request the names of modules to delete when it is run outside of monitor. Up to an 80 character message is accepted from the console which consists of variable length (1 to 8 characters) module names separated by commas. In addition to module names, two special meaning operands may be entered:

$STOP - If this operand is used, it must be the last entry. This operand specifies no input tape. When it is recognized in the parameter string, the program goes to normal termination.

$PRINT- This operand causes a listing of the names of all the modules contained in the OML.

When existing modules are deleted from the OML, all the records used by that module are released for subsequent use. For this reason, no file re-organization is required. When a later version of a module is transcribed to the OML, the module is technically deleted from the file and then re-entered as an addition. This allows the size of the module to change without regard for the original area occupied.

DOMLMR will not properly catalog a module containing a sort due to the fact that OMLU does not convert all of the required information contained on SYSUT1. If an attempt is made to convert and catalog a module containing a sort, OMLU will reflect the omitted information in on error listing, produce an OMF and DOMLMR will catalog OMLU's output without warning. Such a cataloged module will not link properly and should be deleted from the disc OML.

Backup for an OML maintained by DOMLMR is provided by the RCA utility Disc Dump and Reload (DDRL). Once a module is cataloged into the OML, there is no way to retrieve it

except via LNKEDT. There is no facility to extract any particular module from the file for subsequent storage on tape or punched cards.

DOMLMR and PREOML contain a constant which reflects the volume serial number of the disc containing the OML. For this reason, no run time parameters are required for proper device assignment. The only program change required is setting these values which are at a tag 'NAME3.'

The user must bear in mind that subsequent software releases may render certain cataloged modules unusable due to changes to the language translators that modify generated coding. The main area of concern is code generated for FCP and by COBOL for the 'CALL' and 'RETURN' verbs. As changes of this type are seldom, they do not pose a serious problem for DOMLMR users.

The storage capacity is:

70/590 - 4 records per track -
1688 bytes per records
70/564 - 3 records per track -
1129 bytes per record

Cylinder one tracks zero thru eight contain the Object Module Directory - one 16 byte entry for each module contained in the library:

70/590 - 105 entries per record
3778 entries maximum (note 1)
70/564 - 70 entries per record
1888 entries maximum (notes 1 and 2)

Cylinder one, track nine, contains the DOMLMR control record.

Cylinder one, tracks eleven thru nineteen, for 70/590 are not used.

Cylinder two thru the last cylinder in the extent contain the data records or actual modules. Each module will require at least two

of these records for storage. The data records are fixed length of 1688 or 1129 bytes to facilitate update writes to the disc; however, the data contained in them is variable length.

Note 1 - Two of the entries in this area are control entries, the first one and the last one.

Note 2 - The 70/564 does not use the last byte of the 1129 byte directory record.

The PREOML and DOMLMR routines are programmed for both 70/590 and 70/564 support. The device used is determined by the assignment made. No program changes are required.

If DOMLMR terminates abnormally, it cannot be run again until the disc extent has been re-established by DDRL. Access is still possible by LNKEDT.

Equipment:

One tape station for input (SYSUT2).
One disc drive with the 'Object Module Lib' extent.
One printer if the $PRINT option is used in reply to the 'MODULES TO DELETE ?' message.

Memory Required:

Approximately 17,000 bytes.

Source Language:

Assembly

Input Data Format:

The input to DOMLMR is an OML formatted as described in the TOS Utilities Manual 70-35-302. Parameter cards are not required for this program. The names of modules to delete from the disc OML are accepted from the console when the utility is run outside of monitor. The 'Operating Procedures' section of this narrative describes the format and options for the console request.

Output Data Format: The output of DOMLMR is described in appendix A of this narrative. Appendix A also corrolates the input fields that are used with the output format on disc.

Timing: The amount of time required to read the tape and transcribe it to disc.

Remarks:

1. PREOML and DOMLMR are available as source card decks.

2. I would be glad to answer any questions regarding DOMLMR for any interested party.

Operating Procedures (Initialization):

1. Allocate disc area for a file named 'OBJECT MODULE LIB'. This area must begin on cylinder one, track zero, and consist of at least two full cylinders. Only full cylinder allocation is acceptable to the DOMLMR system. The extent must consist of contiguous cylinders.

2. Change the volume serial number in the DC entry called 'NAME3' of both the DOMLMR and PREOML programs to reflect the volume serial numbers allocated in Step 1.

3. Assemble and run the PREOML program. The 'STOP CONDITIONS' portion of this narrative explains the typeouts produced. No parameters are required.

4. Assemble and transcribe the DOMLMR program to the master 'PGMLIB' on the executive disc. The system is now ready for use.

(Extent Change):

From time to time it may be necessary to increase the size of the extent allocated as the OML. This can be accomplished by running RAALLR to de-allocate (not purge) the extent and then to re-establish it. Following the RAALLR run, PREOML must be run to update the control record to reflect the change prior to running DOMLMR. The 'STOP CONDITIONS' portion of this narrative explains the typeouts produced by PREOML. It is recommended that DDRL be run to provide a tape backup for the extent immediately after PREOML has been run.

(General):

DOMLMR may be run as part of a monitor job stream or independently under the TDOS executive. Device assignment is automatic for the required disc device.

INPUT: An Object Module File (OMF) which is SYSUT2 out of the OMLU utility. This assignment is optional - see 'MODULES TO DELETE ?' message.

OUTPUT: Disc resident OMF.
Printer (SYSLST) - optional - See 'MODULES TO DELETE ?' message.

PARAMETERS: No punched card parameters - see 'MODULES TO DELETE ?' message for console parameter format and options.

MESSAGES FROM PREOML

MESSAGE	MEANING	ACTION
EXTENT NOT ALLOCATED IN FULL CYLINDERS	The disc extent established by RAALLR for use with DOMLMR must be allocated in increments of full cylinders. Run RAALLR again to correct the problem and retry PREOML.	None - program terminates.
EXTENT DOES NOT BEGIN AT CYLINDER 1 HEAD 0	The disc extent established by RAALLR for use with DOMLMR must begin at that location. Run RAALLR again to correct the problem and retry PREOML.	None - program terminates.
NEW ENDING CYL NO IS = OR < OLD ENDING CYL NO	The new allocation decreased the size of the extent rather than increasing it, or, the new area was not allocated contiguous to the old area.	None - program terminates. Run RAALLR again to correct the problem and re-run PREOML.
RECORD READ WAS NOT CONTROL RECORD	See 'MESSAGES FROM DOMLMR ROUTINE'	
IS THIS RUN TO CHANGE EXTENTS? (Y OR N)	Self-explanatory.	Response of 'N' causes the entire object module library area to be pre-formatted with dummy records. Response of 'Y' causes pre-formatting of the new area only and updating of the control record.

MESSAGES FROM DOMLMR ROUTINE

MESSAGE	MEANING	ACTION
TAPE MOUNTED IS NOT AN OML	Input to DOMLMR must be an OML (output of OMLU).	None - program terminates. Correct problem and restart the program.
RECORD READ WAS NOT CONTROL RECORD	A control record is maintained in the first cylinder of the 'OBJECT MODULE LIB' extent. A record was read from that area but it is not the control record. Area may not have been pre-formatted by the PREOML routine or an attempt was made to update the master OML with DOMLMR.	None - TERMD
MODULES TO DELETE ?	Program is requesting the name or names of modules to be removed from the OML. This message is typed only if the program is run outside of monitor.	If you do not wish to delete any modules reply 'EOT' else, enter 1 to 8 character module names in any sequence separated by a comma. If you wish to delete modules only and have no input OML, enter $STOP as the last module name. If you wish to print a listing of all module names contained in the library enter $PRINT as a module name. (Continued)

MESSAGE	MEANING	ACTION
MODULES TO DELETE ? ...CONT..		The $PRINT operand may be used along with module names for deletion and/or with the $STOP operand. It is recommended that the $PRINT operand be used as the last operand entered or immediately preceeding the $STOP operand. Up to 80 characters, including commas, may be entered in reply to this message.
MODULE XXXXXXXX NOT IN LIBRARY	XXXXXXXX represents a module name designated to be deleted. The module name was not found in the directory in cylinder 1.	none - program continues
CAUTION ... OVER 95% OF DATA AREA IS USED UP >>>>>>>>	The disc extent must be extended or unused modules must be removed to provide more room.	None - program continues
OUT OF ROOM MODULE XXXXXXXX NOT TRANSCRIBED	While searching the record table, the end of the table was reached without locating an open record.	None - program terminates. Reload program and delete module indicated. Delete additional modules if possible or else extend disc file area.
NUMBER OF UNUSED DATA RECORDS AT START OF THIS RUN IS XXXXX	This message will be typed the first time the program is run in a day. If this figure is low, delete unused modules or	None - program continues.

MESSAGE	MEANING	ACTION
NUMBER OF UNUSED DATA RECORDS AT START OF THIS RUN IS XXXXX (Continued)	extend the file size upon completion of the run. This message is also typed if programs are to be deleted, in which case the message is for information only.	
NUMBER OF UNUSED DATA RECORDS AT END OF DELETION IS XXXXX	Information only. Message is typed after completion of the delete logic. The difference between this message and the previous message will show the number of records made available as a result of the deletion.	None - program continues.
EXTENT HAS CHANGED - RUN PREOML AND RESTART	RAALLR utility was run to change the extent size on disc. PREOML must now be run to update the disc resident control record prior to accessing the file with DOMLMR.	Program terminates - run PREOML and restart.
FILE NOT CLOSED IN PREVIOUS RUN - THIS RUN TERMINATED.	Reconstruct disc extent from DDRL backup and restart program.	Program terminates - No action taken.

MESSAGES FROM DOMLMR AND PREOML	MEANING	ACTION
DISC I/O ERROR - TYPE R FOR RETRY (Note 1)	I/O to disc has been issued twice without success.	Try at least one more time before giving up.
MOUNT VSN XXXXXX - RUN OLC - TYPE C TO CONTINUE	Required volume, as indicated by XXXXXX is not on line.	Self explanatory.
FOLLOWING FILE NOT FOUND (The line following the message will be the 6 byte volume serial number followed by a 44 byte file name.)	The VTOC of the specified disc was searched and the file was not found. PROGRAMMER: Check the length and contents of the &EMTX operand of the GETEM macro. OPERATOR: May have 2 disc packs on line with same volume serial number.	None - TERMD

Note 1 - This message is also used by the program outside of the macro.

Programmer Considerations for Use of the Disc Object Module Library Maintenance Routine (DOMLMR)

The DOMLMR program accepts an Object Module Library (OML) tape as input and transcribes every module on the tape to disc. The output of a program translator (SYSUT1) is an Object Module File (OMF). To convert an OMF to an OML for use with DOMLMR, it is necessary to run the Object Module Library Update (OMLU) routine. The input to OMLU is a SYSUT1 tape, the output is SYSUT2. The following example shows where the OMLU control cards go within a translator job stream. The parameters shown convert all the modules on SYSUT1 to an OML for subsequent transcription to disc by DOMLMR.

Once a module is transcribed to disc, it will remain there until it is deleted via a console message. The program will not request modules for deletion when run under monitor.

When a given module is processed through DOMLMR, the module directory on disc is searched to see if the module already exists. If it does, it is replaced by the new version. If it does not exist, it is added to the file. The only restriction to the use of the DOMLMR is that MODULES CONTAINING A SORT MAY BE TRANSCRIBED TO DISC, BUT THE LINKAGE EDITOR WILL NOT BE ABLE TO LINK THEM BACK. If you do put a sort module out to the library, inform operations so that they may delete it to free up the disc space.

```
// STARTM
// JOB
// PARAM
// translator (ASSMBL, COBOL, FORTRN, RPG)

     SOURCE DECK (MODA)

// translator

     SOURCE DECK (MODB)

// EXEC OMLU
 COPY NONE
 CATALO SYSUT1
 END
// EXEC DOMLMR
// LNKEDT
 PROG MODAB
 INCLUDE SYSOML(MODA,MODB)
// ENDMON
```

MODA and MODB will be cataloged as separate modules. If MODA or MODB or both must be re-translated at a later date, the'// EXEC OMLU' thru '// ENDMON' control cards remain the same. For example, if MODB required re-translation, it would be transcribed to disc, following OMLU, and then linked with the original version of MODA.

Input Record To Output Record Conversion

See TOS Utilities Manual 70-35-302 (Object Module Library) for a description of the input tape.

To illustrate the conversion from tape Object Module Library (OML) format to disc OML format, charts have been intersperced with the narrative. The space between each dot represents two bytes. The tape and disc positions indicated are zero relative. A field with no tape position indicated in the 'From Tape' line indicates information generated by the program. A layout with no 'To Disc' line indicates that the particular record type can be located anywhere within the data bytes alloted in each disc record. The first 8 positions reflect the five position CCHHR address of the next logical record for the module, followed by a three position hexidecimal count field of the number of data bytes that are used.

The basic processing is as follows:

Record type $(00)_{16}$ (Descriptor Block) - fixed length of 16 bytes on disc.
Each 16 byte entry is merged into a sequential location in one of the records located in cylinder 1 track 0 thru 8.
Record type $(01)_{16}$ (Index Block) - starts a new record in the data portion of the file (note 1).
Record type $(02)_{16}$ (Descriptor Block) - starts a new record in the data portion of the file (note 1) and is immediately followed on the output record by the next input transaction $(03)_{16}$, $(04)_{16}$ or $(05)_{16}$.

Note 1: The data portion of the file is cylinder two track zero to end of extent for 70/590 and 70/564. This is the area covered by the track table contained in the DOMLMR control record. Cylinder one tracks zero thru eight contains the Object Module Directory, cylinder one track nine is the DOMLMR control record. Cylinder one tracks eleven thru twenty are not used on the 70/590.

Record Size: 70/590 is 1688 byte records - 4 per track - 1680 data portion.
70/564 is 1129 byte records - 3 per track - 1121 data portion.

Object Module Directory Block $(00)_{16}$

Disc Record Layout

	A	B	C	D	E
From TAPE			9 16		
To DISC	0 4	5 7	8 15	16 20	21 23
Size	5	3	8	5	3

Field Contents

A. CCHHR of next Object Module Directory Block (next sequential record in cylinder one).

B. Three position byte count for this block - represents the sum of the 16 byte entries only. This field is calculated in the Object Module Directory Block Logic $(00)_{16}$.

Fixed length entries of 16 bytes - one per module.

C. Eight position module name.

D. CCHHR of the corresponding Index Block in the data portion of the field. (established when the Index Block - $(01)_{16}$ is read from tape.)

E. Three position displacement of the Index Block within the record specified in field 'D'. (zero in all cases because each Index Block starts a new record)

General:

1. The total record length is 1129 bytes (16 byte entries X 70 entries per record) + 8 control characters + 1 unused byte at the end of the record) for 70/564. For the 70/590 the total record length is 1688 bytes (16 byte entries X 105 entries per record) + 8 control characters.

2. DOMLMR uses tracks 0 thru 8 of cylinder 1 to contain the Object Module Directory.

3. Entries do not overflow records.

4. The first entry of the first record contains - 'OMLU 22 10 YYJJJ' where YYJJJ is the year and julian date of the last reference to the file by DOMLMR.

5. The last entry of the Directory contains a module name of lozenges, a CCHHR of 'CCHHR' and a displacement of hex zero except for 2^7 of the high order position.

Index Block $(01)_{16}$
Disc Record Layout

	A	B	C	D	E	F	G	H	I	J	K
From TAPE				1 1 1 8				2 2 5 7	4 5 7 4	5 5 5 8	3 4 9 6
To DISC	0 4	5 7	8 9	1 1 0 7	1 2 8 2	2 3	2 6	2 2 7 9	3 3 0 7	3 4 8 1	4 4 2 9
Size	5	3	2	8	5	3	1	3	8	4	8

	L	M	N	
From TAPE	3 3 1 8	5 5 7 8	5 7 9 0	7 2 1 9 8
To DISC	5 5 0 7	5 5 8 9	6 7 0 1	7 2
Size	8	2	12	

Index Block $(01)_{16}$
Disc Record Layout

Field Contents

A. CCHHR of first detail record for this module. Same as field 'E'. Established when the $(02)_{16}$ record is read from tape.

B. Byte count for this block - represents size of data portion only.

C. Byte count for the record.

D. Module name.

E. CCHHR of first detail record for this module. (Same as field 'A') Established when $(02)_{16}$ record is read from tape.

F. Displacement of data in the record located at the CCHHR address indicated in field 'E'. (zero in all cases because each Object Module descriptor Block $(02)_{16}$ starts a new record).

G. Unused - pad with zero.

H. Module length.

I. Extern name.

J. Starting address.

K. DDNAME (for include)

L. OMNAME (for include)

M. Unused - set to zero.

N. First 12 byte entry name (8 bytes) and starting address (4 bytes). This field is repeated for each entry, extern and or common for this module.

General:

1. Each Index Block begins a new record.

2. The Index Block is variable length as determined by number of entries, externs and common. (total entries, externs, and common X 12 Bytes per entry + 50).

3. The address of the Index Block record is determined by the table search logic in the TTOA logic.

Object Module Descriptor Block $(02)_{16}$
Disc Record Layout

	A		B		C	D	E	F			G							
From TAPE						0	1	2		5	6							1 3
To DISC	0	4	5	7	8 9	1 0	1 1	1 2		1 5	1 6							2 3
Size	5		3		2	1	1	4			8							

Field Contents

A. CCHHR of next record for this module or hex zero if this is the last record for the module.

B. Block length - initially set to the maximum data bytes to reflect a full record in the $(02)_{16}$ processing. Reset to adjusted length when the next $(02)_{16}$ is read or the end of the input tape is read in the End of Job Logic.

C. Record length - '000E' for $(02)_{16}$ record.

D. Block Code $(02)_{16}$

E. Type of block that follows (See Utilities Manual for code explanation.)

F. Hex zero - reserved for future use.

G. Module name.

General:

1. Each Object module Description Block begins on a new record. It will be immediately followed by one of the other record types $(03_{16}$-$05_{16})$

2. The CCHHR address of this record is contained in the corresponding Index Block $(01)_{16}$ fields A and E.

EXTRN Block $(03)_{16}$

Disc Record Layout

	A	B	C	D	E	F	G
From TAPE		0	1	8 11	12 19	20	22
To DISC							
Size	2	1	1	4	8	1	2

Field Contents

A. Length of $(03)_{16}$ record - starting in tape relative position 12 is a variable number of 11 byte EXTRNS. This length is computed in the 03 logic by multiplying the number of these 11 byte entries by 11 and adding 6 additional bytes for the fixed length portion of the disc record.

B. Block Code - $(03)_{16}$

C. Block Subcode (See Utility Manual for code explanation)

D. Filler - not used.

E. EXTRN

F. Type Code

G. ESID

General

1. Fields E, F and G represent one EXTRN. These fields are repeated for each EXTRN.

2. The EXTRN block may be located anywhere in the data portion of a disc record and may overflow records.

Text Block $(04)_{16}$

Disc Record Layout

	A	B	C	D	E
From TAPE		0	1	4 7	2 0.....N
To DISC					
Size	2	1	1	4	up to 1044

Field Contents

A. Length of $(04)_{16}$ record - this size is calculated by adding 6 bytes to the 'Block Byte Count' field positions 2 and 3 of the $(04)_{16}$ record.

B. Block code $(04)_{16}$

C. Block Subcode (See Utilities Manual for code explanation.)

D. Load address of next block.

E. Text - variable size - up to 1044 bytes.

General

1. The move logic will insure that fields A, B and C will fit on the end of the current output record when field 'A' is moved. To accomplish this the current output record must have 10 or more bytes remaining prior to the 'A' field move. If it does not have sufficient room, the length of the current output record (OTARA + 6 and 7) is modified, the record is written to disc and the 'A' field is placed in the next record starting in OTARA + 8.

2. The 04 record may reside anywhere in the data portion of a disc record and may overflow records.

Modifier Block $(05)_{16}$

Disc Record Layout

	A	B		C	D	E
From TAPE		0	1	4 7	2 3	1 2.....N
To DISC						
Size	2	1	1	4	2	up to 117 char.

Field Contents

A. Length of $(05)_{16}$ record - this size is calculated by subtracting 4 from the length of the tape record read.

B. Block Code $(05)_{16}$

C. Block Subcode (See Utilities Manual for code explanation.)

D. Load address of next block.

E. Modifier count.

F. Modifiers - 10 bytes each. The length of this variable portion of the record is calculated by subtracting 12 from the length of the input record. (See Utilities Manual for contents of modifier record.)

General

1. The 05 record may reside anywhere in the data portion of disc record and may overflow records.

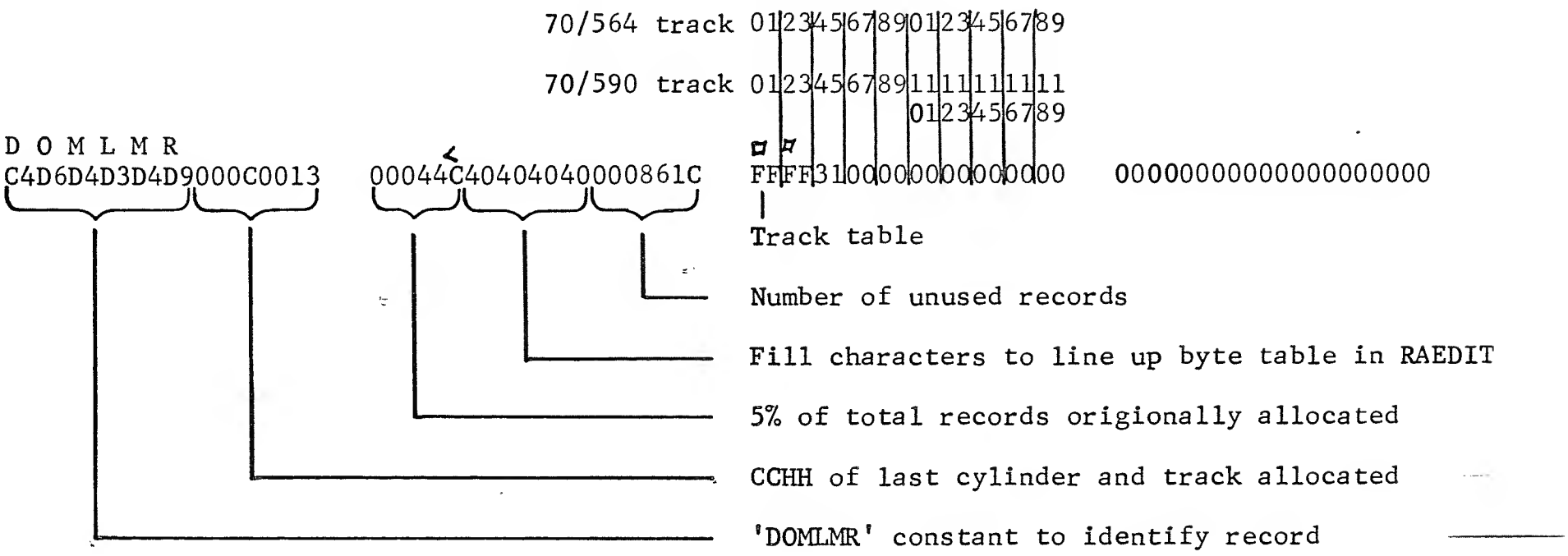
2. Fields A thru E may not be split, the balance of the input record may be.

CONTROL RECORD FORMAT

THIS RECORD IS CONTAINED ON CYLINDER 1, TRACK 9, RECORD 1

FIELD	POSITION	SIZE	DESCRIPTION
1	0-5	6	RECORD IDENTIFIER (DOMLMR)
2	6-9	4	CCHH OF LAST TRACK IN EXTENT
3	10-12	3	5% OF ORIGINAL NUMBER OF RECORDS AVAILABLE. PROGRAM WILL TYPE WARNING MESSAGE WHEN FIELD 5 BECOMES EQUAL TO OR LESS THAN THIS AMOUNT. (PACKED)
4	13-16	4	NOT USED - FOR RAEDIT ALIGNMENT ONLY.
5	17-19	3	NUMBER OF UNUSED RECORDS IN DATA PORTION OF FILE (PACKED).
6	20-2019	2000	ONE BYTE REPRESENTS 2 TRACKS. THE TOP HALF OF THE TYPE (2^4-2^6) REPRESENTS RECORDS 1 THRU 3 (RELATIVE) FOR THE ODD NUMBERED TRACKS. THE BOTTOM HALF OF THE BYTE (2^0-2^2) REPRESENTS RECORDS 1 THRU 3 (RELATIVE) FOR THE EVEN NUMBERED TRACKS. BITS 2^3 AND 2^7 REPRESENT RECORD 4 FOR THE 70/590. A 1 BIT INDICATES RECORD IN USE, A 0 BIT INDICATES RECORD NOT IN USE.

The following example illustrates the DOMLMR control record.



Track Table;
The 'FFFF' in the first two positions indicates that records 1 thru 4 of tracks 0 thru 3 of cylinder 2 are used for a 70/590. The same indication for a 70/564 would be '7777' because only records 1 thru 3 are used.

The '31' indicates:

Track 4

2^7 = 0 -record 4 not used (70/590 only)
2^6 = 0 -record 3 not used
2^5 = 1 -record 2 used
2^4 = 1 -record 1 used

Track 5

2^3 = 0 -record 4 not used (70/590 only)
2^2 = 0 -record 3 not used
2^1 = 0 -record 2 not used
2^0 = 1 -record 1 used

Starting with the table location corresponding to the ending CCHH+1 thru table position 2000 all locations are set to 'FF' for 70/590 or to '77' for 70/564 by PREOML. This is to prevent generating an address beyond the allocated area.
Each decade of the track table represents 1 cylinder for the 70/590. Each half decade represents a cylinder for the 70/564. The table starts at cylinder 2.

The following example illustrates the use of the DOMLMR control record for disc address generation.

A partial RAEDIT of a DOMLMR control record:

```
                                                   ┌──Track table of control record──>
D O M L M R                      <                 ↓
C4D6D4D3D4D9000C0013   0004C40404040000861C   FFFF3100000000000000   00000000000000000000
                                             ┌1 1 1 1 1 1 1 1 1 1   1 1 1 1 1 1 1 1 1 1
                      Sample core locations {0 0 0 0 0 0 0 0 0 0   1 1 1 1 1 1 1 1 1 1
                                             └0 1 2 3 4 5 6 7 8 9   0 1 2 3 4 5 6 7 8 9
```

Address generation is accomplished in the Table to Address (TTOA) logic in DOMLMR as follows:

```
 102 address of the first table position
     that does not contain 'FF' or '77'.
-100 subtract starting address of table.
   2
+  1 add 1.
   3
   3 double the result.
   6
+ (1)this value is -1 if the unused bit
     is located in 2^4 thru 2^7 else zero.
   5
-  1 subtract 1.
   4 = track number - if over 10 or 20,
     divide by 10 or 20 for cylinder num-
     ber, in which case, the remainder will
     be the track number. The result is
     added to the base address of C2, H0
     to arrive at the actual address.
```

Reset unused bit in table based on CCHHR address. This is accomplished in the Address To Table (ATOT) logic in DOMLMR.

```
          C C H H R
Example:  0002000403
         -     2 subtract starting cylinder no.
            0000
        X     20 multiply by tracks per cylinder
                 (10 for 70/564 or 20 for 70/590)
            0000
        +   0004 add the head number.
        2)  0004 divide by 2 tracks per byte in
            0002 table
        +    100 add starting address of table
             102 -effective table address.
```

NOTE Next available Record is determined by a test under mask (TM) starting with 2^4- 2^7 then 2^0- 2^3. Once an open bit is detected, it is turned on by an OI instruction to indicate used. See TTOA logic.
An XC instruction is used in ATOT to reset used bits to zero.